# Relazione Progetto Shop Register

## Descrizione delle funzionalità

Il progetto implementato simula la funzionalità di una cassa generica di un negozio. Esso consente di inserire oppure cancellare degli elementi che vengono acquistati per poi ricevere lo scontrino, il quale può essere restituito in due modi diversi a seconda dell’ordinamento degli elementi desiderato.

Il progetto implementato in Java simula la funzionalità di una cassa generica di un negozio.

Come cassa offre la possibilità di inserire e togliere degli articoli, questi possono essere degli articoli semplici oppure composti da più elementi.

In seguito, abbiamo lo scontrino, il quale può essere uno semplice scontrino oppure uno scontrino di cortesia senza prezzo da accompagnare ad un regalo.

Ed infine, esso può essere restituito in due modi diversi a seconda dell’ordinamento degli elementi.

## Pattern Applicati

I pattern che ho deciso di applicare sono i seguenti:

1. Composite
2. Strategy
3. Decorator

## Struttura progetto

Il progetto è stato diviso in diversi package che andrò a descrivere di seguito.

### item

Dentro questo package troviamo Item e le sue due implementazioni che sono SingleItem ed CompoundItem che rappresentano gli articoli del negozio.

Abbiamo SingleItem che rappresenta un singolo articolo e CompoundItem che rappresenta un articolo composto da una lista di articoli.

Ed è stato qui che ho scelto di implementare il pattern Composite, siccome essendo un negozio esso è composto da articoli singoli/semplici ma anche da articoli che sono un insieme di articoli singoli (come nel caso del kit per universitari di Ikea). E se in futuro si scegliesse di inserire nuovi articoli nel negozio il pattern Composite ne facilita l’operazione.

### sortingStrategy

Questa non è altro che l’implementazione del pattern Strategy, il quale ho scelto di usare per l’ordine usato con cui si andrà a restituire gli articoli. Che a seconda delle esigenze può essere in ordine alfabetico oppure in ordine numerico, ovvero dall’articolo meno caro al più caro.

Ho fatto la scelta di questo pattern siccome mi permette di variare la modalità di stampa dello scontrino.

### Bill

Da modificare completamente questa parte

Dentro questo package troviamo l’implementazione dell’oggetto scontrino, il quale in seguito verrà restituito dalla classe Register che si trova nel package shopRegister.

Qui è presente l’implementazione del pattern Decorator, siccome quando avviene la stampa di uno scontrino si parte inizialmente da una stringa vuota sulla quale si vanno ad aggiungere sempre più cose partendo in questo caso dagli elementi che sono stati ordinati dalla classe scelta del SortingStrategy trasformati in una stringa alla quale è stato concatenato alla fine il prezzo totale degli articoli.

Lo scontrino decorato con tutti i dettagli viene restituito dalla classe GreetingsBillDecorator la quale aggiunge alla fine della stringa gli auguri finali in fondo allo scontrino.

### shopRegister

Qui è presente la classe Register la quale rappresenta il registratore di cassa.

Che ci permette di inserire e togliere gli articoli da una lista, ottenere il totale e lo scontrino.

All’interno di questa classe ho scelto di passare la strategia con la quale ordinare la Lista di articoli quando viene chiesto di stampare lo scontrino e non altrove, siccome secondo me la scelta dell’ordinamento non fa parte dell’astrazione della cassa.

In seguito, ho preferito usare List a Comparable siccome non avrei potuto inserire più elementi uguali, siccome il set del Comparable non lo permette.

### Test

Invece per quanto riguarda i test ho scelto di usare AssertJ siccome mi permette di scrivere i test in maniera chiara e leggibile, oltre al fatto che mi è più facile testare le Liste siccome le ho usate all’interno del progetto.

Tutti i test presenti testano tutte le classi che al loro interno hanno un ragionamento, al cui interno troviamo delle operazioni.